資料結構與程式設計
期末專題報告

Functionally Reduced And-Inverter Graph (FRAIG)

系所：資訊管理學系
年級：三年級
姓名：黃亮穎
學號：B05705017
聯絡電話：0928754541

目錄

壹、資料結構設計・・・・・・・・・・・・・・・・・・・3

1. cirMgr
2. cirGate
3. myHashMap

貳、演算法・・・・・・・・・・・・・・・・・・・・・・4

1. CirSweep
2. CirOptimize
3. CirStrash
4. CirSimulation
5. CirFraig

參、綜合比較・・・・・・・・・・・・・・・・・・・・・6

肆、課程建議與心得・・・・・・・・・・・・・・・・・・7

壹、資料結構設計

1. cirMgr：

Data member：

_header：將.aag檔裡不同種類的gate的數量存進struct，看起來整齊，也方便維護目前不同種類gate的數量，在cirp -pi, -po等指令實作上較不易出錯。

CirGate ** _gateList：大小為m+o+1的CirGate*動態陣列，讀檔有讀到的gate就會new出pointer，否則為0，好處是在之後處理gate連接或刪除的時候只要判斷CirGate*是否為零，避免對不必要的gate做動作而造成segmentation fault。

GateList dfsList：一個vector<cirGate*>存放Depth First Search會traversal到的gate。

Member function：

buildConnection( )：read circuit完要將相連的gate的fanin、fanout串接，因為要重複做所以將這個動作寫成function。

dfs( )：做完optimize, strash, fraig後會優化circuit，改變dfsList內容，因此要呼叫dfs( )，將vector清空再重traversal一次。

dfsTravel( )：屬於recursive function，每個gate的fanin做完dfsTravel( )才會換自己。

2. cirGate：

child class PI, PO, AIG, CONST0, UNDEF繼承cirGate。

Data member：

有gateID，有vector<unsigned>, vector<cirGate*> 各兩個分別存放這個gate的fanin(只有兩個), fanout(可能很多個)，並且因為我是存variable gateID，所以要多存一個vector<bool>存fanin有沒有invert。

為了實作function可以比較精簡且直觀，所以才會gateID與cirGate*都存，但若著重節省記憶體使用量的話其實也可以不存vector<unsigned>。但兩個都存有一個麻煩點是在做optimize時串接fanin, fanout要同時對兩個vector操作，這可能也是我的程式比reference慢的原因之一。

bool used ：這個變數若是true，代表gate有在dfsList裡面，設計原因是sweep時要去除不在dfsList裡面的gate，可以用它來判斷。

Member function：

printGate( ), getTypeStr( )：不同種類的gate有不同輸出結果所以寫在child class。

addFanin( ), addFanout( ), addInv( )：將vector push_back( )這個動作包起來使程式比較易懂。

faninRecursive( ), fanoutRecursive( )：實作report fanin, fanout的helper function，recursive的去print比較直觀且符合要求格式。

3. myHashMap：

實作strash時需要使用，Hash的key我是簡單的將兩個fanin的literal ID相乘起來，設計key的考量完全就是因為簡單。可能沒有hash的很平均，但對於只做到strash的我來說，效能上的影響與reference相比還能接受（綜合比較會再敘述）。

使用b05705043 資管三 唐瑋廷的hw6的code完成final project。

貳、演算法

1. CirSweep

對所有gate，我先判斷PI不能被sweep，利用上面提到的used判斷gate是否在dfsList裡，不在的話就要sweep掉，並且要維護還在dfsList裡面的gate的fanin, fanout，避免cirp, cirg時segmentation fault。最後將sweep掉的gate的cirGate* 歸還給記憶體(delete)，cirGate* 設成0，將這個gateType對應到_header的數量減少(ex. if sweep a AIGgate, - -_header.a)。

2. CirOptimize

一個AIGgate的兩個fanin若：

存在const0(false)，gate的output必定是0
存在const1(true)，gate的output必定是1
identical，gate的output就是fanin的output
inverted，gate的output就是0

以上，gate不影響結果，所以可以去掉。

所以我直觀地將optimize分成四種case，一樣做串接然後刪掉多餘的gate，更新_header。在串接fanin, fanout時原本是直接做取代，但是有幾個case，fanin, fanout數量會改變，就只能以push_back, erase更新vector<unsigned>, vector<cirGate*>，後來覺得這樣很亂，若當初有考慮到此情況的話應該直接都使用push_back跟erase。

3. CirStrash

兩個擁有完全相同fanins的gate可視為一樣，因此可以將一個移除。至於判斷是否一樣若直接判斷兩個fanin會是O(n^2)，所以設計一個hash key(literal id of fanin1 * literal id of fanin2) ，若key一樣gate才可能一樣，將複雜度直接降到O(n)。

對所有dfsList裡的AIGgate，若不在hash裡面就放進去，在的話就執行刪除串接，方式與optimize相似，最後一樣要歸還記憶體與更新_header。

4. CirSimulation

對PI做simulate，一次會simulate sizeof(size_t) * 8 組patterns，做完simulation每個gate會有simValue，一開始所有gate會在一個fec group裡，若simValue不同就可以將兩個gate分成不同組，fec group存法是使用hashMap。simulation可分成對所有gate或event-driven的，event-driven能夠提升效率，但在演算法設計上也變複雜，因此一開始是以all gate的方式去實作。

5. CirFraig

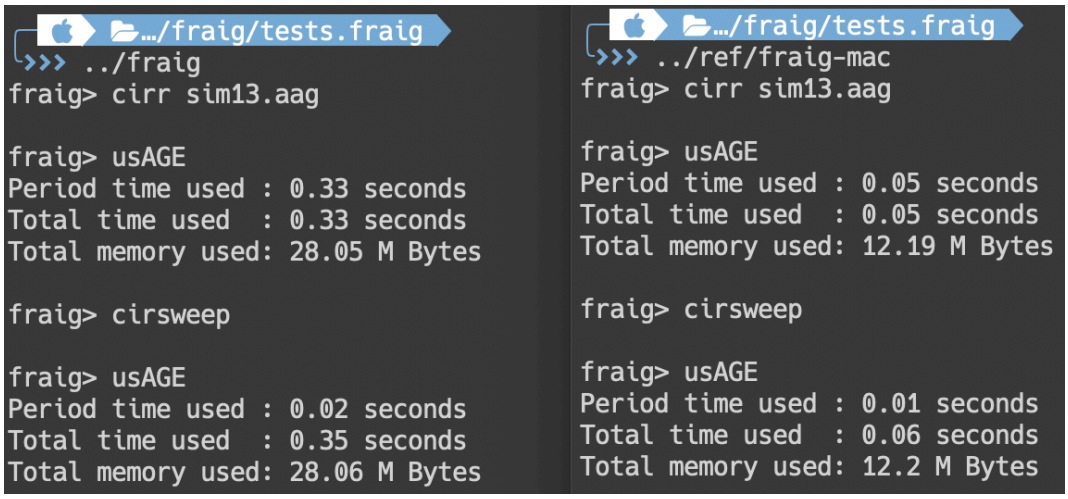
Functionally Equivalent Candidate (FEC) 代表這些被分在同一組的gate還只是候選，可能他們事實上還是不一樣，只是經過simulation後simValue還一樣，因此要用SAT去證明是否真的是functionally equivalent，equivalent則可以取代。

參、綜合比較

因為其他檔案太小看不太出差別，所以測 sim13.aag

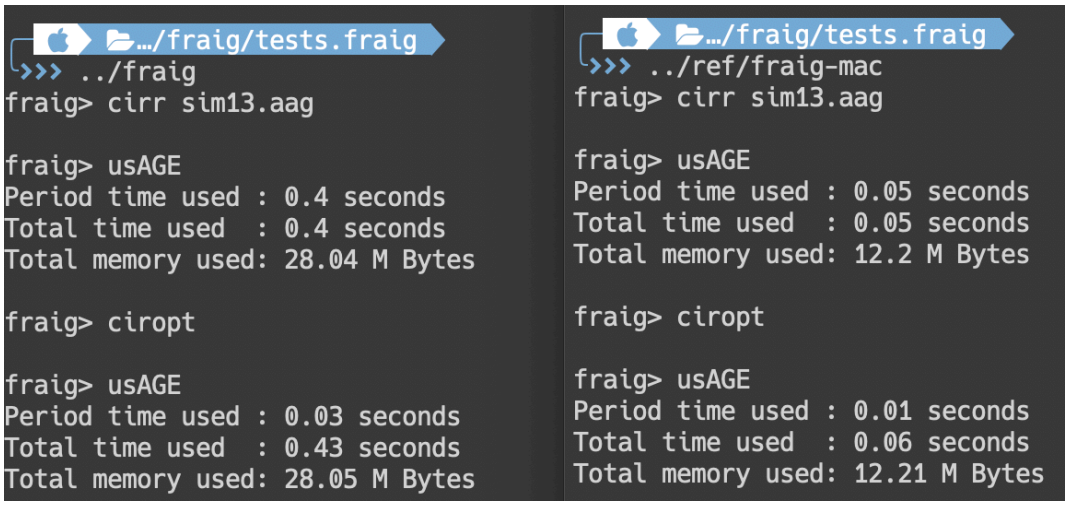
sweep效率比較：

(mine/ref)



從圖可以看出 sweep 演算法所花的時間和 ref 差不多，但空間上多了一倍，推測是因為我gateID和CirGate*都有開vector存fanin, fanout，因此多了 ref 所使用的空間不少。

optimize效率比較：

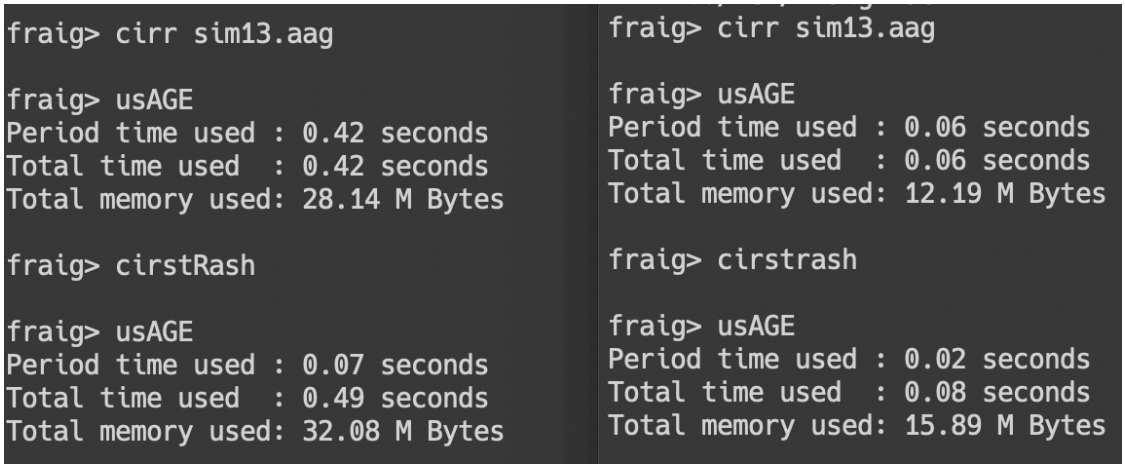


(mine/ref)

時間一樣差不多，空間原因與sweep會是一樣的。

strash效率比較：

(mine/ref)



時間會是reference的三倍，我覺得是hash key設計的不好的緣故，導致在hash table裡面query的時候花了較多時間。設計一個更好的key使gate存的越分散應該能提升效能。

肆、課程建議與心得

課程建議：

寫final的時候發現，有時候程式有bug但給的範例測資測不到bug，希望有時候測資可以再特殊一點，不然像我這種code寫頗亂想重砍掉又怕時間來不及的人就常常有小bug會出錯。邏輯處理與debug當然是自己的工作，但如果有更好的測資會輕鬆許多。

心得：

之所以想修這門課是因為我二上以前很混，所以系上程設（大一下）、資結（大二上）都沒有很認真上，又在聽到去年有修課的兩位系上同學說DSNP很硬但收穫很豐富，就像趁機會惡補自己的資訊基礎。學期初的時候，真的是連class都快忘記要怎麼用了，但整個學期過完，不敢說自己已經摸熟C++與資料結構了，但是與學期初的我相比，我敢肯定的告訴自己真的有進步許多，很謝謝有DSNP這門課讓我可以重新學習程設、資結，謝謝Ric老師有耐心的幫我們複習C++，講解各種資料結構，還不時的突然放梗將全班逗得哈哈大笑的，要不是教室很悶我一定不會有幾節課不小心睡著（咦～）。最令我覺得老師很努力的地方是竟然半夜兩、三點了還會解答FB社團裡面的問題！！！當然也謝謝助教幫忙批改作業與解決作業上的問題，謝謝一起修課一起討論的同學們。